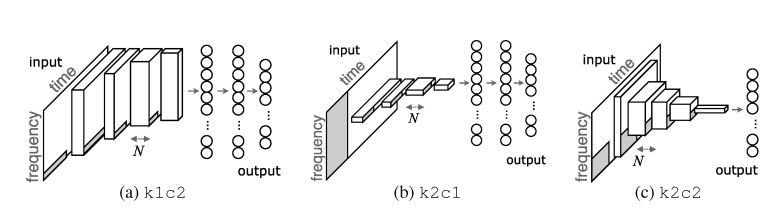
相关论文：

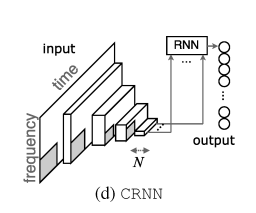
1. Classifying environmental sounds using image recognition networks
2. Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection
3. Convolutional Recurrent Neural Networks for Music Classification

采用模型：

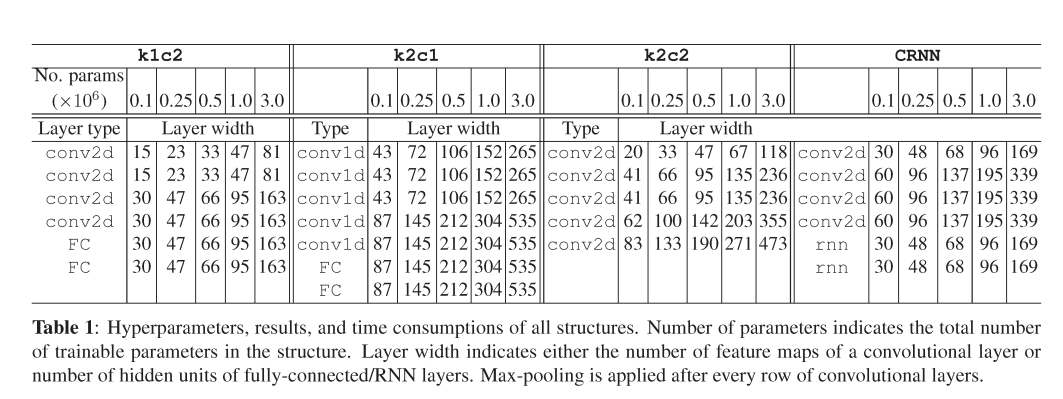
CNN



CRNN



相关参数：



模型修改：

对模型结构进行了微调，相关结构如下：

CNN:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv1 (Conv2D) (None, 126, 1249, 20) 200

batch\_normalization (BatchN (None, 126, 1249, 20) 80

ormalization)

pool1 (MaxPooling2D) (None, 63, 312, 20) 0

dropout1 (Dropout) (None, 63, 312, 20) 0

conv2 (Conv2D) (None, 61, 310, 41) 7421

batch\_normalization\_1 (Batc (None, 61, 310, 41) 164

hNormalization)

pool2 (MaxPooling2D) (None, 30, 77, 41) 0

dropout2 (Dropout) (None, 30, 77, 41) 0

conv3 (Conv2D) (None, 28, 75, 41) 15170

batch\_normalization\_2 (Batc (None, 28, 75, 41) 164

hNormalization)

pool3 (MaxPooling2D) (None, 14, 18, 41) 0

dropout3 (Dropout) (None, 14, 18, 41) 0

conv4 (Conv2D) (None, 12, 16, 62) 22940

batch\_normalization\_3 (Batc (None, 12, 16, 62) 248

hNormalization)

pool4 (MaxPooling2D) (None, 3, 4, 62) 0

dropout4 (Dropout) (None, 3, 4, 62) 0

flatten (Flatten) (None, 744) 0

Linear1 (Dense) (None, 256) 190720

Linear2 (Dense) (None, 50) 12850

activation (Activation) (None, 50) 0

=================================================================

Total params: 249,957

Trainable params: 249,629

Non-trainable params: 328

CRNN:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv1 (Conv2D) (None, 126, 1249, 30) 300

batch\_normalization (BatchN (None, 126, 1249, 30) 120

ormalization)

pool1 (MaxPooling2D) (None, 42, 624, 30) 0

dropout1 (Dropout) (None, 42, 624, 30) 0

conv2 (Conv2D) (None, 40, 622, 60) 16260

batch\_normalization\_1 (Batc (None, 40, 622, 60) 240

hNormalization)

pool2 (MaxPooling2D) (None, 20, 311, 60) 0

dropout2 (Dropout) (None, 20, 311, 60) 0

conv3 (Conv2D) (None, 18, 309, 60) 32460

batch\_normalization\_2 (Batc (None, 18, 309, 60) 240

hNormalization)

pool3 (MaxPooling2D) (None, 6, 154, 60) 0

dropout3 (Dropout) (None, 6, 154, 60) 0

conv4 (Conv2D) (None, 4, 152, 60) 32460

batch\_normalization\_3 (Batc (None, 4, 152, 60) 240

hNormalization)

pool4 (MaxPooling2D) (None, 1, 50, 60) 0

dropout4 (Dropout) (None, 1, 50, 60) 0

permute (Permute) (None, 60, 1, 50) 0

reshape (Reshape) (None, 50, 60) 0

LSTM1 (LSTM) (None, 50, 60) 29040

LSTM2 (LSTM) (None, 60) 29040

dropout5 (Dropout) (None, 60) 0

Linear (Dense) (None, 50) 3050

activation (Activation) (None, 50) 0

=================================================================

Total params: 143,450

Trainable params: 143,030

Non-trainable params: 420

数据集：

ESC50数据集 [karolpiczak/ESC-50: ESC-50: Dataset for Environmental Sound Classification (github.com)](https://github.com/karolpiczak/ESC-50)

ESC-50 数据集是由 2000 个环境音频录音组成的标注集，适用于环境声音分类方法的基准测试。该数据集由 5 秒钟长的录音组成，分为 50 个语义类别（每个类别有 40 个示例）。

对5s的音频进行分类，输入原始音频的mel-频谱，输出为音频类型。对于2000条数据，选择其中400条作为测试集，1600条作为训练集，为了增加训练集的条数，对原始训练集进行了扩展，主要包括以下方法：

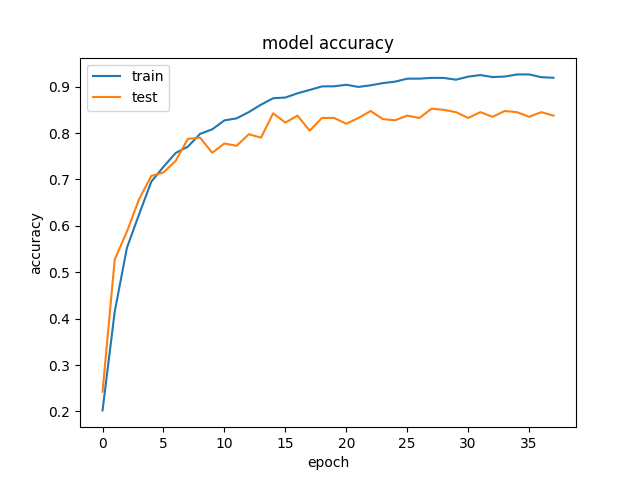
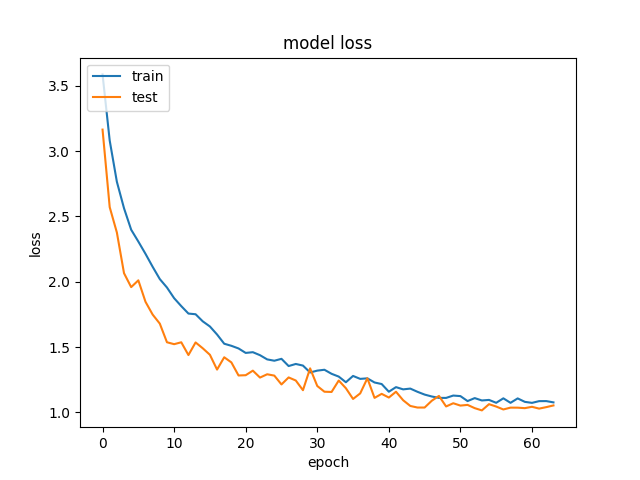
1.数据增强（Data Augmentation）：对现有的音频样本进行一系列的变换和处理，例如随机添加白噪音、音频拉伸、音频平移等。这样可以产生更多的变体，扩展数据集的规模。

2.合成数据（Data Synthesis）：使用现有的音频样本合成新的样本。可以通过将不同的音频片段进行拼接、叠加或混音等方式来生成新的音频数据。

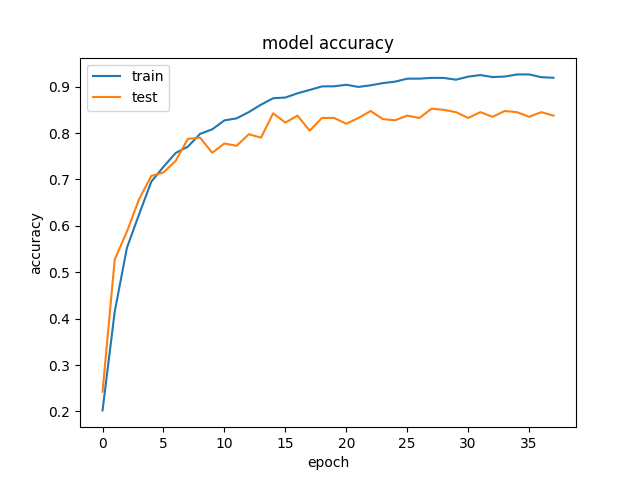
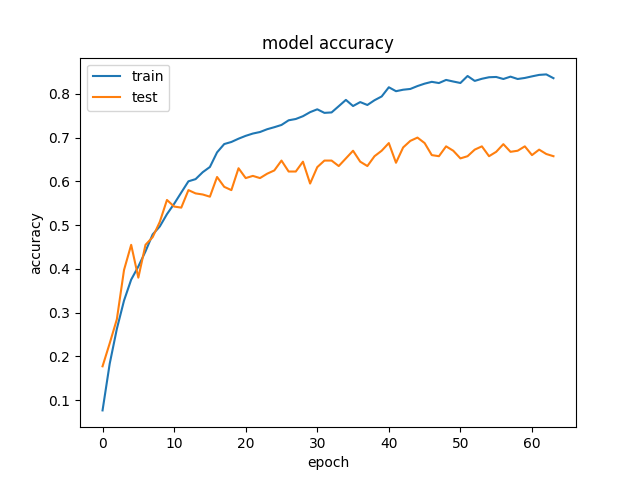
扩展之后的训练集有6400音频数据，输入模型，进行训练，10轮验证集loss未下降停止训练，每3轮验证集loss未改善调整学习率为原来的0.6倍，初始学习率0.001。

训练过程：

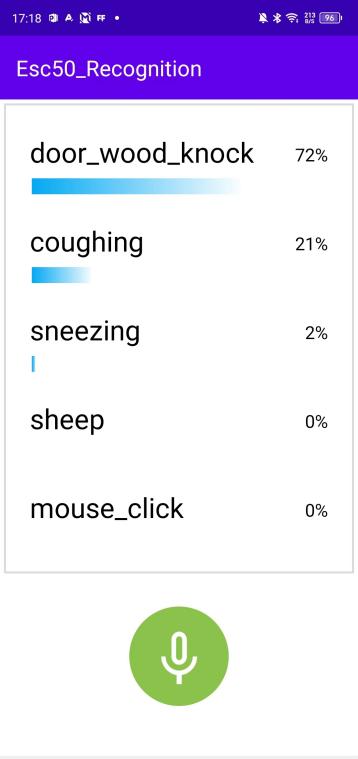
CNN:



CRNN：



基于TFLITE进行音频识别的App主界面：



模型压缩方法：

量化方法1：动态量化，训练后量化最简单的形式是仅将权重从浮点静态量化为整数（具有 8 位准确度）推断时，权重从 8 位准确度转换为浮点，并使用浮点内核进行计算。此转换会完成一次并缓存，以减少延迟。为了进一步改善延迟，“动态范围”算子会根据激活的范围将其动态量化为 8 位，并使用 8 位权重和激活执行计算。此优化提供的延迟接近全定点推断。但是，输出仍使用浮点进行存储，因此使用动态范围算子的加速小于全定点计算。

量化方法2：混合量化，通过确保所有模型数学均为整数量化，进一步改善延迟，减少峰值内存用量。对于全整数量化，需要校准或估算模型中所有浮点张量的范围，即 (min, max)。

与权重和偏差等常量张量不同，模型输入、激活（中间层的输出）和模型输出等变量张量不能校准，除非我们运行几个推断周期。因此，转换器需要一个有代表性的数据集来校准它们。这个数据集可以是训练数据或验证数据的一个小子集（大约 100-500 个样本）。对模型进行全整数量化，但在模型没有整数实现时使用浮点算子（以确保转换顺利进行）。为了与原始的全浮点模型具有相同的接口，此 tflite\_quant\_model 不兼容仅支持整数的设备（如 8 位微控制器）和加速器（如 Coral Edge TPU），因为输入和输出仍为浮点。

量化方法3：Float16量化 2x smaller, GPU acceleration CPU, GPU将权重量化为 float16（16 位浮点数的 IEEE 标准）来缩减浮点模型的大小。float16 量化的优点如下：将模型的大小缩减一半（因为所有权重都变成其原始大小的一半）。实现最小的准确率损失。支持可直接对 float16 数据进行运算的部分委托（例如 GPU 委托），从而使执行速度比 float32 计算更快。float16 量化的缺点如下：它不像对定点数学进行量化那样减少那么多延迟。默认情况下，float16 量化模型在 CPU 上运行时会将权重值“反量化”为 float32。（请注意，GPU 委托不会执行此反量化，因为它可以对 float16 数据进行运算。）

剪枝方法1：

对于CNN来说，剪枝方法1是对BatchNormalization以及Conv2D层进行剪枝，剪枝的方法为修剪开始的稀疏度为0.5，修剪结束的稀疏度为0.8，微调训练20轮，每轮进行剪枝操作；

对于CRNN来说，剪枝方法1也是对BatchNormalization以及Conv2D层进行剪枝，剪枝的方法为修剪开始的稀疏度为0.5，修剪结束的稀疏度为0.8，微调训练20轮，每轮进行剪枝操作；

剪枝方法2：

对于CNN来说，剪枝方法2是对Dense层进行剪枝，剪枝的方法为修剪开始的稀疏度为0.5，修剪结束的稀疏度为0.8，微调训练20轮，每轮进行剪枝操作；

对于CRNN来说，剪枝方法2是对Dense层以及LSTM层进行剪枝，剪枝的方法为修剪开始的稀疏度为0.5，修剪结束的稀疏度为0.8，微调训练20轮，每轮进行剪枝操作；

模型压缩实验结果：

本次实验的实验测试平台在传音手机TECNO CK9n移动设备上进行，相关的参数如下：

|  |  |
| --- | --- |
| 相关平台 | 名称 |
| 手机型号 | TECON CK9n |
| Android版本 | Android 13 |
| 操作系统版本 | HiOS v13.0.0 |
| 内存大小 | 8GB |
| 存储大小 | 512GB |
| CPU版本 | 联发科天玑Dimensity 8050 |
| CPU核心数目 | 8 |

CNN原始模型：

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为124.9ms/次 |
| 部署到APP上的内存开销 | 推理时APP内存开销约为165.4MB： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到29%左右： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗8.10mAh： |
| 模型在测试集上的推理指标 | Accuracy：85.25% |
| 模型文件大小.tflite | 981KB |
| 模型压缩文件大小.zip | 903KB |

CNN动态量化模型

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为118.1ms/次，提升5.4% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为157.4MB，减少4.8%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到27%左右，降低6.9%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.25mAh，降低10.5%： |
| 模型在测试集上的推理准确率 | Accuracy：85.0%，准确度相比原来下降0.3% |
| 模型文件大小.tflite | 256KB，减少73.9% |
| 模型压缩文件大小.zip | 220KB，减少75.6% |

CNN混合量化模型

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为124.2ms/次，提升0.6% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为143.0MB，减少13.5%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到27%左右，降低6.9%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.13mAh，降低12.0%： |
| 模型在测试集上的推理准确率 | Accuracy：85.5%，准确度无下降，甚至相比原来略微上升0.3% |
| 模型文件大小.tflite | 256KB，减少73.9% |
| 模型压缩文件大小.zip | 218KB，减少75.9% |

CNN Float16量化模型

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为121.9ms/次，提升2.4% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为157.5MB，减少4.8%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到27%左右，降低6.9%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗6.96mAh，降低14.1%： |
| 模型在测试集上的推理准确率 | Accuracy：85.25%，准确度无下降 |
| 模型文件大小.tflite | 496KB，减少49.4% |
| 模型压缩文件大小.zip | 451KB，减少50.1% |

CNN剪枝模型（BatchNormalization + Conv2D）

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为124.4ms/次，提升0.4% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为146.2MB，减少11.6%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销仍然达到29%左右，无明显降低： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.21mAh，降低11.0%： |
| 模型在测试集上的推理准确率 | Accuracy：78.5%，准确度相比原来下降7.9% |
| 模型文件大小.tflite | 852KB，减少13.1% |
| 模型压缩文件大小.zip | 782KB，减少13.3% |

CNN剪枝模型（Dense）

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为120.0ms/次，提升3.9% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为145.0MB，减少12.3%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到27%左右，降低6.9%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.41mAh，降低8.5%： |
| 模型在测试集上的推理准确率 | Accuracy：75.5%，准确度相比原来下降11.4% |
| 模型文件大小.tflite | 424KB，减少56.8% |
| 模型压缩文件大小.zip | 357KB，减少60.5% |

CRNN原始模型

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理10次计算单次推理平均时间为155.7ms/次 |
| 部署到APP上的内存开销 | 推理时APP内存开销约为202.5MB： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到30%左右： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗8.49mAh： |
| 模型在测试集上的推理准确率 | Accuracy：68.0% |
| 模型文件大小.tflite | 582KB |
| 模型压缩文件大小.zip | 528KB |

CRNN动态量化模型

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为127.8ms/次，提升17.9% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为181.3MB，减少10.5%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到29%左右，降低3.3%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.13mAh，降低16.0%： |
| 模型在测试集上的推理准确率 | Accuracy：67.5%，准确度相比原来下降0.7% |
| 模型文件大小.tflite | 171KB，减少70.6% |
| 模型压缩文件大小.zip | 141KB，减少73.3% |

CRNN混合量化模型

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为152.0ms/次，提升2.4% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为157.8MB，减少22.1%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到28%左右，降低6.7%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.18mAh，降低15.4%： |
| 模型在测试集上的推理准确率 | Accuracy：66.75%，准确度相比原来下降1.8% |
| 模型文件大小.tflite | 173KB，减少70.3% |
| 模型压缩文件大小.zip | 137KB，减少74.1% |

CRNN Float16量化模型

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为150.9ms/次，提升3.1% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为200.1MB，减少1.1%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到29%左右，降低3.3%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.62mAh，降低10.2%： |
| 模型在测试集上的推理准确率 | Accuracy：68%，准确度无下降 |
| 模型文件大小.tflite | 305KB，减少47.6% |
| 模型压缩文件大小.zip | 269KB，减少49.1% |

CRNN剪枝模型（BatchNormalization + Conv2D）

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为147.9ms/次，提升5.0% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为195.9MB，减少3.3%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到27%左右，降低10%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.45mAh，降低12.2%： |
| 模型在测试集上的推理准确率 | Accuracy：62.5%，准确度相比原来下降8.1% |
| 模型文件大小.tflite | 348KB，减少40.2% |
| 模型压缩文件大小.zip | 302KB，减少42.8% |

CRNN剪枝模型（Dense + LSTM）

|  |  |
| --- | --- |
| **性能指标** | **性能测试值** |
| 单次推理时间（一段长为5s的音频，采样率为32000，不含预处理） | 推理20次计算单次推理平均时间为144.4ms/次，提升7.3% |
| 部署到APP上的内存开销 | 推理时APP内存开销约为194.1MB，减少4.1%： |
| CPU开销（单次推理峰值CPU） | 单次推理时峰值CPU开销达到27%左右，降低10%： |
| 单次推理功耗开销 | 平均单次推理功耗开销较小，峰值的Energy消耗为Light，持续采样推理5分钟(60次推理加上5分钟音频采集)大概消耗7.24mAh，降低14.7%： |
| 模型在测试集上的推理准确率 | Accuracy：59.75%，准确度相比原来下降12.1% |
| 模型文件大小.tflite | 406KB，减少30.2% |
| 模型压缩文件大小.zip | 360KB，减少31.8% |